Smart Exploration in Reinforcement Learning using Absolute Temporal Difference Errors

Clement Gehring
School of Computer Science
McGill University
Montreal, QC, Canada
clement.gehring@mail.mcgill.ca

Doina Precup
School of Computer Science
McGill University
Montreal, QC, Canada
dprecup@cs.mcgill.ca

ABSTRACT

Exploration is still one of the crucial problems in reinforcement learning, especially for agents acting in safety-critical situations. We propose a new directed exploration method, based on a notion of state controlability. Intuitively, if an agent wants to stay safe, it should seek out states where the effects of its actions are easier to predict; we call such states more controllable. Our main contribution is a new notion of controlability, computed directly from temporal-difference errors. Unlike other existing approaches of this type, our method scales linearly with the number of state features, and is directly applicable to function approximation. Our method converges to correct values in the policy evaluation setting. We also demonstrate significantly faster learning when this exploration strategy is used in large control problems.

Categories and Subject Descriptors

F.8 [**Theory of computation**]: Reinforcement learning

Keywords

reinforcement learning, exploration, temporal-difference error

1. INTRODUCTION

Autonomous agents are confronted with the problem of making decisions in the face of uncertain and incomplete information. A standard framework for modelling this type of problem is reinforcement learning [16, 18], in which agents learn, from a stream of data, how to choose actions in order to maximize their long-term return. A crucial problem in reinforcement learning, which has spurred extensive research, is the *exploration-exploitation trade-off*: how should an agent act in order to balance its ability to find out new information, versus exploiting knowledge that it already has. One of the most flexible and successful approaches is *directed exploration*, which gives an agent bonuses in order to encourage it to visit new states, or take new actions. Some of these approaches are heuristic, e.g. [19, 14, 8]. For other approaches, theoretical guarantees exist. For example, the "optimism in the face of uncertainty" strategy [6, 2, 15] assumes that unknown parts of the state space will be very rewarding, and uses bonuses based on the standard deviation of the observed value estimates. Under these conditions, sample complexity bounds can be established both for model-based and model-free algorithms. While this approach is natural to implement in small environments, where the value function can be represented by a table, it does not generalize easily to large problems, where function approximation is necessary. Recent work [9, 5] has extended these ideas to continuous state spaces; however, these methods either rely on a batch of data, so they are not incremental, or they build a model of the environment, which leads to quadratic complexity in terms of the number of features. This is prohibitive in domains with high-dimensional observations, such as robot sensor data or images. Also, certain popular approximators, such as tile coding, generate very large feature spaces, for which quadratic complexity in the number of features is not acceptable.

In this paper, we propose a simple way of measuring how uncertain (or equivalently, how controllable) a state is. Intuitively, from the point of view of value estimation, if a particular state (or state-action pair) yields a lot of variability in the temporal-difference error signal, it is less controllable. We use this quantity as an added bonus in the exploration process. Note that, depending on the tolerance to risk, one can in principle either encourage or discourage exploration of less controllable state-action pairs. We approach this problem from a safety perspective, so the goal of our method is to encourage the agent to seek controllable regions of the environment. This idea has an interesting side effect: it reduces as much as possible the noise in the value function updates, which leads to faster and more stable convergence.

We use the mean absolute deviation of the temporal difference (TD) errors as a measure of controlability. As described later on, this measure is more robust to noise than the standard deviation [3, 20], so it is more adequate for the reinforcement learning setup. This measure was used with success by [12, 22] in order to construct new value function features, as well as hierarchies of abstract behaviours. Here, we focus instead on using it to bias the action choices.

The paper is organized as follows. Section 2 contains definitions and notation. In Section 3 we define our notion of controlability, explain how it is learned from data, present its use in exploration, and discuss the theoretical guarantees of the algorithm. Section 4 presents a small grid-world illustration, to build some intuition about the behaviour of the algorithm. Section 5 contains a partially observable example, illustrating the robustness of the approach to violations

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

of usual reinforcement learning assumptions. In Section 6 we present an application to helicopter control, a complex, high-dimensional benchmark provided as part of the RL library[1]. Our proposed method provides significantly faster learning, and much better policies. Finally, Section 7 contains a discussion and describes avenues for future work.

2. BACKGROUND

We assume the usual reinforcement learning framework, in which an agent interacts with its environment at discrete time steps $t = 1, 2, \ldots$. At each time step, the agent observes the state of the environment, $s_t \in S$, and chooses an action $a_t \in A$. One time step later, the agent receives a reward r_{t+1} and observes a new state s_{t+1}. In a Markovian environment, both r_{t+1} and s_{t+1} depend only on s_t and a_t. For every state-action pair $s \in S, a \in A$, $R(s, a)$ is the expected value of the reward that will be received when a is taken in s, and $P(\cdot|s, a)$ is the next state distribution.

A stochastic Markovian policy $\pi : S \times A \to \mathbb{R}$ defines a probability distribution for actions in each state:

$$\pi(s, a) = Pr(a_t = a|s_t = s).$$

This choice of action is typically driven by a *value function*, which estimates the expected long-term return of each state or state-action pair. We mainly focus on the state-action value function, $Q^\pi : S \times A \to \mathbb{R}$, defined as:

$$Q^\pi(s, a) = \mathbf{E}_\pi \left[r_{t+1} + \gamma r_{t+2} + \ldots |s_t = s, a_t = a\right]$$

where $\gamma \in (0, 1)$ is a discount factor, used to emphasize rewards received earlier over those received in the future. For a Markovian problem in which the state and action spaces are discrete and small enough, Q^π can be represented by a table with one entry for each (s, a) pair. The values can be learned incrementally by the following update:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t \tag{1}$$

where Q denotes the estimate of Q^π, $\alpha \in (0, 1)$ is the learning rate, and

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{2}$$

is the *temporal-difference (TD) error* produced as a result of the transition. Note that both a_t and a_{t+1} are assumed to be chosen according to π. It has been shown [4] that this update process converges to correct value estimates in the limit, under standard stochastic approximation conditions. Hence, $\lim_{t\to\infty} \mathbf{E}_\pi(\delta_t) = 0$.

If the state space is very large or continuous, function approximation is used to represent Q. Perhaps the most common choice is to use a linear approximator of the following form:

$$Q(s_t, a_t) = \theta_{a_t}^T \phi_{s_t} \tag{3}$$

where θ_{a_t} is a parameter vector (with one such vector being learned for each action), and ϕ_{s_t} is a feature vector corresponding to the current state. Then, equation (1) becomes an update to the parameter vector:

$$\theta_{a_t} \leftarrow \theta_{a_t} + \alpha\delta_t\phi_{s_t} \tag{4}$$

where δ_t is still computed according to (2).

The goal of reinforcement learning is to obtain an *optimal policy*, π^*, which has maximal value in all states. In order to obtain such a policy incrementally from data, two main approaches can be used. On-policy algorithms evaluate the policy being followed; hence, the policy has to be mostly greedy with respect to the value function estimates. For example, the Sarsa algorithm [13, 16] uses Equations (1) (or (3)) and (2), but at each step, the policy π is modified according to the current Q estimates. An ϵ-greedy strategy is frequently used to generate the policy used by the Sarsa algorithm. This strategy can be defined as follows. Suppose n actions are available in state s_t, and the maximal value of Q is attained for m of them. Then:

$$\pi(s_t, a_t) = \begin{cases} (1-\epsilon)/m & \text{if } a_t \in \arg\max_a Q(s_t, a) \\ \epsilon/(n-m) & \text{otherwise} \end{cases}$$

Off-policy learning algorithms attempt to find the optimal policy and value function directly, regardless of the policy being followed. The most popular algorithm of this type, Q-learning [21], uses the following TD error signal:

$$\delta_t = r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$$

Exploration methods aim to improve the procedure for choosing actions. In particular, we are interested in *directed exploration*, in which actions are chosen greedily with respect to a function of the form:

$$Q(s_t, a_t) + \omega B(s_t, a_t)$$

where $B(s_t, a_t)$ is a bonus that typically encourages visiting unknown areas. For example, the bonus could be proportional to the time since a particular state-action pair was experienced [19], or it can be proportional to the standard deviation of the action-value function, as done in the class of optimistic exploration methods [2, 15]. However, in applications in which safety is critical, the approach of seeking high-variance areas may not be advisable, as it can lead to negative consequences. Moreover, in the long run, if an area has high randomness in the rewards or transitions, visiting it repeatedly is wasteful, in the sense that additional data will not improve value estimates. If anything, the noise generated in such areas may slow down the convergence of the value function. This is the intuition behind the approach we are about to present.

3. PROPOSED APPROACH

We propose to use the mean absolute temporal-difference error as an indication of how "controllable" a state is. In statistics, there are many ways to measure the spread of a random variable about its mean. Most methods focus on variance and standard deviation, because these measures work well when data comes from a normal distribution, and allow for convenient mathematical manipulation. However, if the data is very noisy, e.g. containing outliers, or violates normality assumptions, empirical estimates of the variance are quite sensitive, and may be dominated by the noise. Huber [3] showed that the mean absolute deviation is a lower bound on the standard deviation, and that it is more robust to noise and normality violations.

For reinforcement learning, it is clear that TD-errors are very noisy, because they are based on estimates of the value function, which are constantly changing. Moreover, in a control problem, the policy is also changing, so one would expect the TD-errors to be non-stationary. Hence, we adopt the mean absolute deviation as a measure of uncertainty

[1]http://library.rl-community.org/wiki/Helicopter(Java)

in this work. Note that even though the expected value of the TD-errors converges to 0, the expected value of the absolute TD-errors will not be 0, unless the environment is deterministic. In this case, of course, one sample from each state-action pair is sufficient, so the exploration problem becomes moot.

We define the controlability of a state-action pair, given a fixed policy π, as:

$$C^{\pi}(s,a) = -\mathbf{E}_{\pi}[|\delta_t||s_t = s, a_t = a] \quad (5)$$

Note that the higher variability of the TD-errors is in a given state, the lower the controlability will be. Controllability can be estimated using the TD-errors in a straightforward way. At each time step t,

$$C(s_t,a_t) \leftarrow C(s_t,a_t) - \alpha'(|\delta_t| + C(s_t,a_t)) \quad (6)$$

where $\alpha' = \beta\alpha$ is a learning rate, with $\beta < 1$. The reason for this setup is that we would like the values to converge quicker, so the estimates of the errors are not as noisy. A similar approach is taken in [17], for computing a very similar signal, but which is used for corrections in off-policy learning. We note that this update uses the TD error which has already been computed anyway, so the additional effort per update is $O(1)$.

If function approximation needs to be used, the controlability can be estimated using a parameter vector, $\mathbf{w}$, similarly to the value function. The parametrization of C should be the same as for Q, and the update to the parameter vector $\mathbf{w}$ of C becomes (by analogy with (4)):

$$\mathbf{w}_{a_t} \leftarrow \mathbf{w}_{a_t} - \alpha'(|\delta_t| + \mathbf{w}_{a_t}^T \phi_{s_t})\phi_{s_t}$$

If the policy π is fixed and Q^{π} is estimated correctly, controlability as given by (5) is well-defined, and (6) provides a consistent estimator for it (though, according to work in statistics, the estimator is biased). However, controlability has to be learned at the same time as the value function estimate. This process (including function approximation) can be shown to converge by a two-time scale analysis, using the ODE method of [1]. For this approach, we rewrite the two iterations (for Q and C) as one iteration, with a combined parameter vector containing both the θs and the $\mathbf{w}$s, then apply Theorem 2.2 of [1]. The analysis is analogous to that of [17], and is left out for clarity of exposition.

The exploration algorithm uses controlability as an exploration bonus, picking actions greedily according to:

$$Q(s_t,a_t) + \omega C(s_t,a_t) \quad (7)$$

where ω is a parameter used to trade off between the desire to obtain high returns, and the minimization of the TD error signals. If $\omega = 0$, the algorithm relies only on the values. If ω is high, the emphasis is on visiting areas of high controlability. This approach can be viewed as an attempt to "regularize" the behavior of the system, by bringing it to trajectories of lower variability.

The idea of restricting the policy space to "safe" policies has been explored before in [10], However, that approach is quite different, as it involves identifying Lyapunov functions for a given domain, and designing policies that are guaranteed to improve the value of such a function. Lyapunov functions do not always exist, and this process may be difficult for a system designer. Our approach is significantly easier to use in practice.

Note that if desired, the controlability bonus can also be used together with a random exploration factor; for example, an ϵ-greedy policy with respect to (7) can be used. This ensures sufficient exploration of all actions.

When the policy changes during the estimation process, obtaining theoretical guarantees requires the policy to change smoothly and slowly compared to the estimates in (7); the contribution of controlability would also have to decrease over time, in order to achieve optimality according to the value function, in the limit. This can be achieved, e.g. using Boltzmann exploration and an approach similar to [11]. However, this strategy is not practically useful, so we do not explore it further.

4. GRID-WORLD EXPERIMENTS

In this section, we aim to provide some intuition on the behaviour of the algorithm, using a simple 18×18 grid-world environment depicted in Figure 1. The red circle represents a fixed goal state which the agent must reach. Furthermore, there are several 'slippery' rectangular patches (shaded in the left panel) which span several states. At every state, the agent must choose from four deterministic actions: up, down, right, left. Each transition from a normal state causes the agent to receive a reward of -1. If the agent transits out of a 'slippery' state, the reward is uniformly distributed in the interval $[-12, 10]$. If the agent attempts to move out of the grid-world, the state stays unchanged and it receives a reward of -10. An episode starts with the agent randomly placed in the grid and stops when either the agent reaches the goal, or 150 steps have been taken without success. We used $\gamma = 1$ and $\epsilon = 0.1$, and we varied the learning rate α and the controlability parameter ω. We ran 10000 trials, each consisting of 2400 consecutive episodes.

The right panel in Figure 1 shows a heat map of the controlability values. Blue indicates high controlability (low TD errors), while hot colors indicate large TD errors. As can be seen, the agent identifies the problematic patches correctly, without any problem. Figure 2 shows the average return and standard deviation on the error bars for Sarsa compared to the exploration approach using controlability. Note that the use of controlability leads to faster learning, but more importantly, it leads to tighter error bars than using plain Sarsa. As expected, using controlability guides the algorithm towards safer parts of the environment. This behaviour is consistent over multiple values of the learning rate.

Figure 3 provides a summary of the standard deviation of the learned value function at the end of the training, as a function of ω (x-axis) and of the learning rate α (different curves). For all values of α, we set $\beta = 0.1$. Note that for all values of α, intermediate values of the controlability parameter perform best.

5. PARTIALLY OBSERVABLE DOMAIN

In this section, we evaluate the robustness of the proposed algorithm to violations of the Markovian assumption. This is important to study as function approximation essentially amounts to introducing partial observability into a system.

We use the simple partially observable Markov Decision Process depicted in Figure 4. The agent always starts in the state labelled 1 and has to travel to the rightmost goal state, following one of two paths. The states are aliased, and

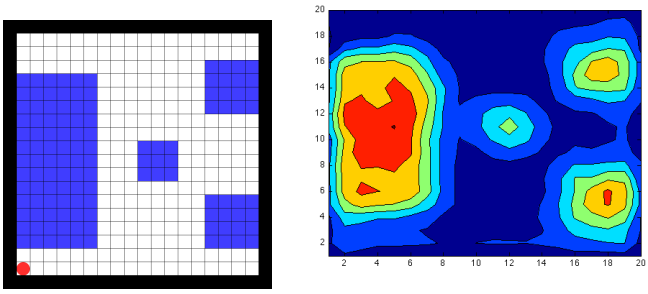

Figure 1: Grid-world environment (left) and learned controlability heat map (right)

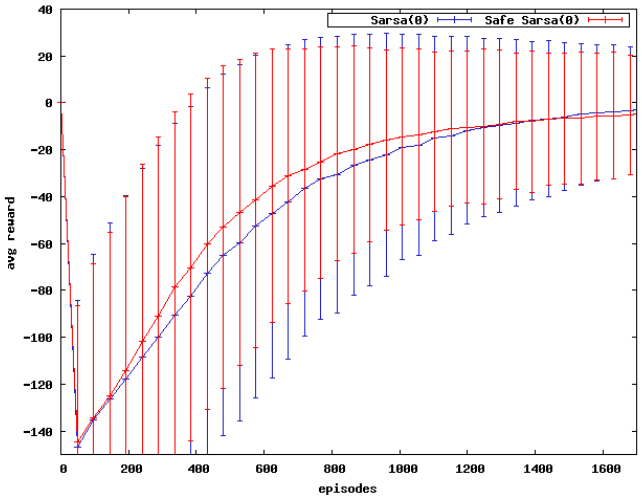

Figure 2: Return of usual exploration (blue) and controlability-based exploration (red) in the grid-world problem

the number in each state encodes the observation that the agent will receive. At each time step, the agent receives a reward of -1, and the actions are deterministic, and depicted by the arrows in the Figure 4. Both paths are identical, except that one path has a state that generates a unique observation. The goal of the experiment was to show both that our algorithm is stable, and that it would have a bias toward the most observable path.

We set $\gamma = 1$ and $\epsilon = 0.1$. Because of the partial observability, using just the immediate value of the controlability is not sufficient, as the update is non-Markovian. We deal with this problem by computing a controlability which looks ahead to the next states:

$$C(s_t, a_t) \leftarrow C(s_t, a_t) - \alpha'(C(s_t, a_t) + |\delta_t| - \gamma_2 C(s_{t+1}, a_{t+1}))$$

where γ_2 is a second discount factor used for controlability. We also use eligibility traces, controlled as usual by parameter λ, to provide faster propagation of information along trajectories. For each pair of λ and γ_2, we optimized α and picked the best value.

Figure 5 presents the fraction of times the observable path was taken. Note that, as expected, the versions using $\lambda = 0.6$ are better, as eligibility traces help overcome the partial observability. However, controlability provides

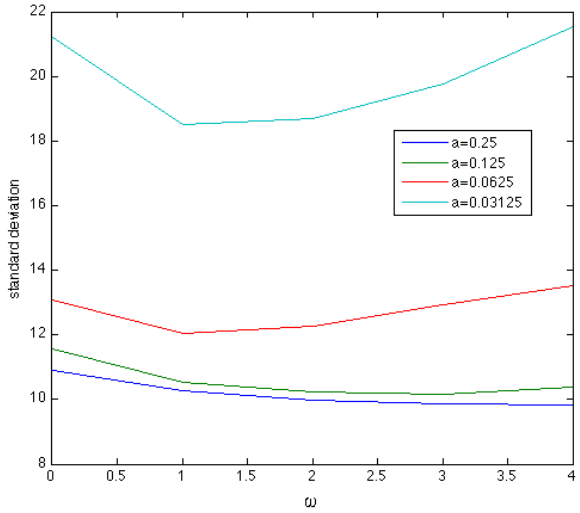

Figure 3: Standard deviation at the end of learning, for different combinations of values for parameters α and ω. Using controlability consistently produces more stable results, and the best value of the parameter ω is not influenced much by the learning rate α

faster learning even in this case, helping the agent latch onto the observable path faster (red curve in Figure 5). For $\lambda = 0$, using controlability in the exploration process makes learning significantly faster and more stable.

Figure 6 uses $\lambda = 0.6$ and $\gamma_2 = 0$, and studies the effect of ω (α has been optimized for each ω setting.) As anticipated, using some controlability ($\omega = 0.5$; green curve) shows improvement over using none. However, a very large emphasis on controlability is problematic, and leads to slower learning, because it prevents the agent from exploring sufficiently.

This example shows that our approach has great potential for domains like robotics, where the state information is inaccessible and all the agent is given are noisy observations.

6. HELICOPTER TASK

In this section, we tackle the complex, high-dimensional problem of controlling a helicopter. The domain was contributed by Pieter Abeel to the Reinforcement Learning Library, and some previous versions were used in reinforcement learning competitions.

The Task

The problem consists of controlling a helicopter so that it hovers over a point of interest. The state information consists of a 12 dimensional vector containing the position (x, y, z), the velocity (u, v, w), the angular velocity (p, q, r) and the orientation (i, j, k) of the helicopter. In order to fly, the agent needs to decide the power level for both the main rotor and the tail rotor, and the orientation of the main rotor. Hence, the action space is 4-dimensional, and all action values are between -1 and 1. We discretized the action space into 256 actions, allowing us to map each dimension to values in the set $\{-0.25, -0.05, 0.05, 0.25\}$. To make the task harder, random winds perturb the helicopter while it is fly-

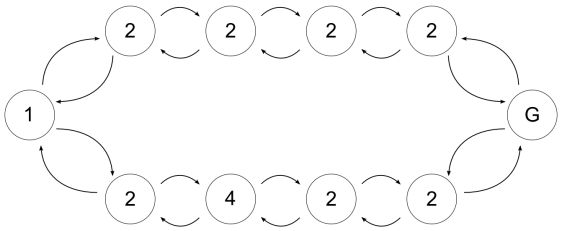

Figure 4: A simple POMDP domain. The arrows represent actions and the states are labelled with the observation received. Note that several states generate the same observation

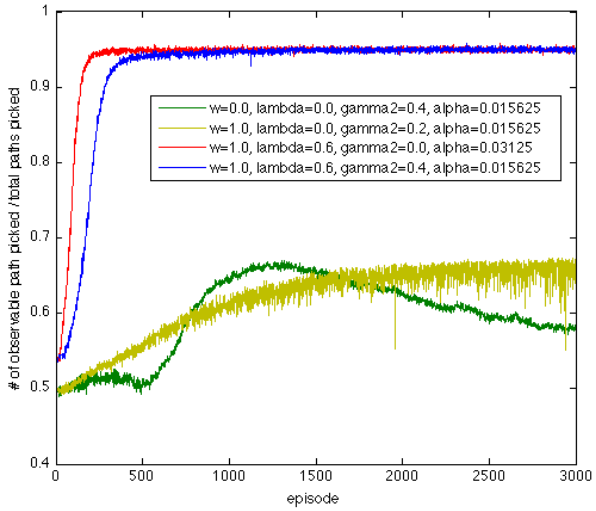

Figure 5: Fraction of time the most observable path was taken, as a function of learning episodes. Lower curves are for $\lambda = 0$, higher ones for $\lambda = 0.6$. The observable state is 4th on the path.

ing. A negative reward proportional to how early the trial is terminated is awarded when the helicopter crashes, or when it drifts away from the hovering area. In addition to that, a penalty proportional to the distance from the hover point is given at each time-step, to encourage the correct behaviour.

We used standard tile coding as the function approximator. Each dimension of the state space was tiled with 96 grids of size 4, resulting in a feature vector of 4608 entries. This configuration was optimized to obtain as good a solution as possible. The reinforcement learning algorithm was Sarsa. We optimized the learning rate for the algorithm which uses no controlability ($\omega = 0$). We then fixed these parameters for the other versions of the algorithm ($\omega > 0$). We performed 50 independent runs for each algorithm.

Results

Figure 7 presents the duration of the episodes, as a function of the number of learning trials. The main challenge

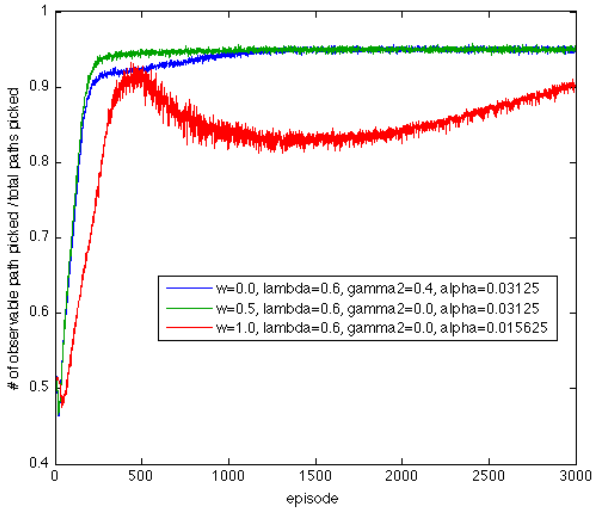

Figure 6: Effect of ω on the success of controlability-based exploration in the POMDP domain. This graph was generated from a POMDP with the fully observable state second on the path.

in the helicopter task is that bad actions are very difficult to identify. The agent could behave perfectly until the last time-step, when a bad action can lead to penalizing otherwise perfect behaviour; or, conversely, the agent could be choosing poor actions, bringing the helicopter in an unstable state, then perform the best possible action to recover, but still crash, penalizing the last good action. As can be seen in Figure 7, controlability is very successful in guiding the agent towards states that are easier to learn. Both versions with $\omega > 0$ are significantly better than $\omega = 0$.

However, modifying the behaviour in this way is not without cost. Preferring 'easy' states might push the agent away from the optimal solution. This is why in Figure 7 we notice a drastic improvement in the learning phase when using controlability very aggressively ($\omega = 50$) but this leads to behavior which is too cautious, does not explore enough, and the final solution ends up being sub-optimal. In future work, it would be worth decreasing ω slowly over time, to allow the behaviour to shift towards the optimal solution.

Reasons of improvement

Intuitively, we would think that risk-taking is a good heuristic for exploring an unknown environment (as evidenced by the sample complexity results for optimistic exploration). This is not the case, however, for the helicopter task. In this domain, taking risk can be very detrimental, since the gaps between good and bad actions are big. It only takes one bad action to crash, but many good ones to fly.

Another interesting aspect in this domain is that states are not isolated - there are typically many sequences of actions that can lead to the same state. This means that the agent can afford to wait and not take a risky or unknown action right away, because it can come back to the same situation a different time, without too much difficulty.

Controllability reflects the variance in the returns due to environment stochasticity, but also the variance caused by

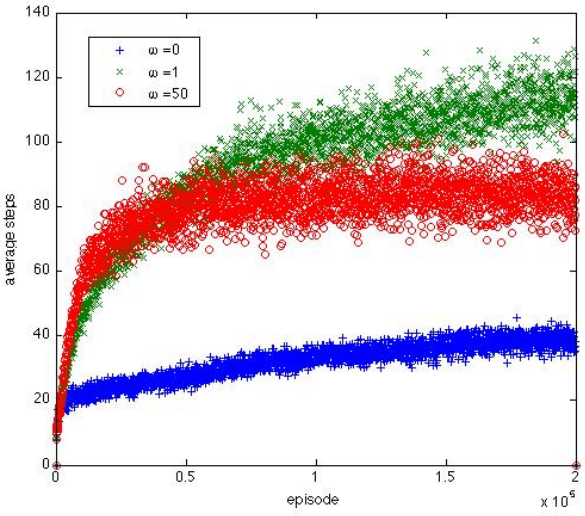

Figure 7: Average time of flight versus the number of episodes. For all three plots, $\alpha_1 = 0.02$, $\alpha' = 0.002$, $\gamma = 1$ and $\epsilon = 0.05$. These values were optimized for $\omega = 0$

agent's poorly learned values. As the agent learns better estimates of $Q(s, a)$, the controlability increases, until it only reflects the variance in the return. This causes the agent to explore first actions that lead to stable or learnt states. Cautious behaviour could slow down learning, but in a unforgiving domain such as the helicopter task, cautious behaviour allows for better survival, which provides more opportunities to explore and learn.

7. DISCUSSION

In this paper, we proposed to use the mean absolute TD error in order to measure the controlability of state-action pairs. This approach proved very useful in speeding up learning, and generating policies that are more robust to detrimental environment effects. We note that controlability could also prove useful in creating a good function approximation representation, by suggesting new features that seek to make the environment more predictable. The results described in Section 5 suggest that this approach could be very useful. The work of [22] is a step in this direction as well.

A similar goal of improving the safety of policies is shared by work on robust MDPs, and mean-variance optimization in MDPs (see e.g., [7]). We note, however, that their approach proposes a joint criterion involving both the mean and the variance of the value function, which is significantly more expensive to optimize. Moreover, in that work, the value function is not learned separately, so optimal values and policies cannot be recovered, even if desired at a later time. We sought to maintain the accuracy of the value function; the controlability only affects the action choices, but the optimal values are not polluted by controlability values.

In our experiments, we kept the parameter ω at a fixed value. However, in general this parameter could be adjusted during learning. In particular, if the application is not safety-critical, one could decrease ω over time, in order to better emphasize the value function.

Future work will focus on gaining more practical experience with this approach, especially in robotics and other safety-critical domains. We are interested in establishing bounds on the sample complexity of the proposed approach. We are also exploring the use of controlability in creating variable resolution tile coding approximators.

Acknowledgements

This research was funded by the NSERC USRA and NSERC Discovery programs. The authors thank Rich Sutton and the anonymous reviewers for useful comments on this work.

8. REFERENCES

[1] V. S. Borkar and S. P. Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control And Optimization*, 38(2):447–469, 2000.

[2] R. I. Brafman and M. Tennenholtz. R-max, a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, pages 213–231, 2002.

[3] P. J. Huber. *Robust statistics*. Wiley, 1981.

[4] T. Jaakkola, M. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

[5] N. K. Jong and P. Stone. Model-based function approximation for reinforcement learning. In *AAMAS*, 2007.

[6] M. J. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

[7] S. Mannor and J. Tsitsiklis. Mean-variance optimization in Markov Decision Processes. In *ICML*, 2011.

[8] N. Meuleau and P. Bourgine. Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2):117–154, 1999.

[9] A. Nouri and M. L. Littman. Multi-resolution exploration in continuous spaces. In *NIPS*, 2009.

[10] T. J. Perkins and A. G. Barto. Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3:803–832, 2002.

[11] T. J. Perkins and D. Precup. A convergent form of approximate policy iteration. In *NIPS*, 2003.

[12] M. Ring and T. Schaul. Q-error as a selection mechanism in modular reinforcement-learning systems. In *IJCAI*, 2011.

[13] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, Cambridge University, 1994.

[14] J. Schmidhuber. Adaptive confidence and adaptive curiosity. Technical Report FKI-149-91, Technische Universitat Munchen, 1991.

[15] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. PAC model-free reinforcement learning. In *ICML*, 2006.

[16] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[17] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvari, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML*, 2009.

[18] C. Szepesvari. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.

[19] S. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, School of Computer Science, Carnegie Mellon University, 1992.

[20] J. W. Tukey. A survey of sampling from contaminated distributions. In *Contributions to Probability and Statistics*, pages 448–485. Stanford University Press, 1960.

[21] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[22] S. Yi, F. Gomez, M. Ring, and J. Schmidhuber. Incremental basis construction from temporal difference error. In *ICML*, 2011.